IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Vertes et al.**	§	Group Art Unit: **2113**
	§	
Serial No.: **10/522,897**	§	Examiner: **Riad, Amine**
	§	
Filed: **February 1, 2005**	§	Confirmation No.: **8427**
	§	
For: **Functional Continuity by Replicating a Software Application in a Multi-Computer Architecture**	§	Attorney Docket No.: **FR920050802US1**

35525
PATENT TRADEMARK OFFICE
CUSTOMER NUMBER

Commissioner for Patents
P.O. Box 1450
Alexandria VA 22313-1450

AMENDMENT PURSUANT TO REQUEST FOR CONTINUING EXAMINATION

Sir:

A fee of $810.00 is required for filing a Request for Continued Examination. Please charge this fee to Yee & Associates, P.C. Deposit Account No. 50-3157. No additional fees are believed to be necessary. If, however, any additional fees are required, I authorize the Commissioner to charge these fees to Yee & Associates, P.C. Deposit Account No. 50-3157.

A two-month extension of time is believed to be necessary. I authorize the Commissioner to charge the one-month extension fee of $460.00 to Yee & Associates, P.C. Deposit Account No. 50-3157. No additional extension of time is believed to be necessary. If, however, an additional extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to Yee & Associates, P.C. Deposit Account No. 50-3157.

In response to the Final Office Action of March 19, 2008, please amend the above-identified application as follows:

Amendments to the Claims are reflected in the listing of claims that begins on page 2 of this paper.

Remarks/Arguments begin on page 7 of this paper.

Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1.-39. (Cancelled)

40. (Currently Amended) A computer implemented method for replicating a software application ~~in a multi-computer architecture cluster~~, the computer implemented method comprising:

executing the software application on a ~~first computer of the cluster forming a~~ primary node to form a master application;

identifying resources and dependencies required by the master application to form required resources;

updating the required resources dynamically on the primary node;

generating a structure of the master application and a dynamic graph of the required resources from the required resources;

replicating the resources by transferring the structure to a set of secondary nodes via a network to form a replica; wherein the set of secondary nodes comprises one or more secondary nodes; ~~associated with the software application on at least one other computer of the cluster forming a secondary node, the resources associated with the software application on the secondary node comprising a virtual memory and calling stack of each process affected, system resources comprising inter-process communication, network connection, and data written on disks; and~~

restoring the replica on the set of secondary nodes to form a set of clone software applications, wherein the set of clone software applications comprises one or more clone software applications;

executing the set of clone software applications on the set of secondary nodes, without loss of context; and

updating the ~~replicated resources~~ set of clone software applications with incremental updates ~~incrementally, using a dynamic introspection mechanism supplying a structure of the software application to be replicated and also supplying a dynamic graph~~ of the required resources of the master application to create a hot standby application. ~~and dependencies implemented of the~~ .

41. (Previously Presented) The computer implemented method according to claim 40, wherein replicating the resources further comprises:

creating and maintaining a dependency tree, based on the dynamic graph, supplying, at all times, information on the replicated resources.

42. (Currently Amended) The computer implemented method according to claim 40, wherein replicating the resources ~~associated with the software application~~ further comprises:

checkpointing the resources on ~~at least one~~ the set of secondary nodes, wherein the checkpointing having an adjustable period.

43. (Currently Amended) The computer implemented method according to claim 42, wherein replicating the resources ~~associated with the software application~~ further comprises:

capturing the resources on the primary node to create captured required resources;

transferring the captured required resources over the network to ~~at least one~~ the set of secondary nodes; and

restoring the captured required resources on the ~~at least one~~ set of secondary nodes.

44. (Previously Presented) The computer implemented method according to claim 42, wherein replicating the resources further comprises:

optimizing the checkpointing.

45. (Previously Presented) The computer implemented method according to claim 44, wherein the checkpointing is incremental.

46. (Previously Presented) The computer implemented method according to claim 44, wherein the checkpointing is discriminating.

47. (Previously Presented) The computer implemented method according to claim 42, wherein the checkpointing further comprises at least one of the following:

processing a synchronization barrier;

managing resources;

managing system resources; and

managing process resources.

48. (Previously Presented) The computer implemented method according to claim 40, wherein replicating the resources further comprises:

replicating applicative data files between the primary node, whereon the software application is run, and a stand-by node.

49. (Previously Presented) The computer implemented method according to claim 40, wherein replicating the resources further comprises:

ensuring functional continuity of the software application in a multi-computer architecture cluster, the software application being executed at a given time on one of the computers of the cluster, called the primary node, while other computers of the cluster are called a set of secondary nodes, wherein ensuring functional continuity further comprises:

replicating the software application on at least one of the secondary nodes to provide ~~at least one~~ a set of clones of the application, wherein the set of clones comprises one or more clones;

updating the ~~at least one~~ set of clones, and

responsive to detecting an event affecting the primary node, switching from the software application being executed on the primary node, to the software application being executed on the ~~at least one~~ set of clones.

50. (Previously Presented) The computer implemented method according to claim 49, wherein replicating the software application is of a holistic nature.

51. (Previously Presented) The computer implemented method according to claim 49, wherein updating the ~~at least one~~ set of clones further comprises updating the set of clones of the application.

52. (Previously Presented) The computer implemented method according to claim 49, wherein ensuring functional continuity further comprises supervising a state of the resources necessary to operate the software application.

53. (Previously Presented) The computer implemented method according to claim 49, wherein detecting an event affecting the primary node further comprises:

responsive to detecting an event affecting the primary node, electing a clone to be substituted for the primary node of the software application, wherein the secondary node on which the clone elect is installed becomes a new primary node.

54. (Previously Presented) The computer implemented method according to claim 53, wherein replicating the resources further comprises:

recording, on the ~~at least one~~ set of clones, messages received by the primary node, the messages being injected into the clone elected as the new primary node when switching.

55. (Previously Presented) The computer implemented method according to claim 40, wherein replicating the resources further comprises:

optimization of information processing resources by load sharing and dynamic process distribution.

56. (Previously Presented) The computer implemented method according to claim 40, wherein replicating the resources further comprises:

performing non-interruptive maintenance by process relocation upon request, over a data-processing resource network.

57. (Previously Presented) The computer implemented method according to claim 40, wherein replicating the resources further comprises:

preserving applicative context in a mobile application.

58. (Previously Presented) A multi-computer system for ensuring functional continuity, capable of running, on at least one computer, at least one software application, the multi-computer system comprising:

a memory comprising a set of instructions;

a processor connected to the memory, capable of executing the set of instructions to implement a method comprising:

ensuring functional continuity of the software application in a multi-computer architecture cluster, the software application being executed at a given time on one of the computers of the cluster, called a primary node, while other computers of the cluster are called a set of secondary nodes, wherein ensuring functional continuity further comprises:

replicating the software application on ~~at least one of~~ the set of secondary nodes to provide ~~at least one~~ a set of clones of the application, wherein replicating the software application is of a holistic nature;

updating the ~~at least one~~ set of clones, and responsive to detecting an event affecting the primary node, switching from the software application being executed on the primary node, to the software application being executed on the ~~at least one~~ set of clones.

REMARKS/ARGUMENTS

Claims 40-58 are pending in the present application. Claims 1-39 are canceled. Claims 40, 42, and 43 are amended. Support for the amendments may be found in the description on page 3 lines 5-6 and lines 19-23, page 6 lines 12-16, page 7 lines 1-2, page 8 lines 18-19, page 12 lines 9-13, page 13 lines 15-34, and page 14 lines 1-9. Reconsideration of the claims is respectfully requested.

I. 35 U.S.C. § 102, Anticipation

The examiner has rejected claims 40-58 under 35 U.S.C. § 102 as being anticipated by Kelkar et al., Cluster Failover for Storage Management Services, (U.S. Patent 7,058,846), (hereinafter "*Kelkar*").

The Examiner states:

> In regard to claim 40
>
> Kelkar discloses a method for replicating a software application in a multi-computer architecture (cluster), whereas said software application may be executed beforehand on a first computer of said cluster forming a primary node and intended for replication on at least one other computer of said cluster forming a secondary node,
> comprising a replication of the resources associated with said software application, characterized in that the replicated resources include: (Figure 2) and (abstract) [This figure shows two nodes 110A and 110B]
> - the virtual memory of each process affected as well as its calling stack,
>
> - system resources (inter-process communication, network connection, etc.) and
>
> - data written on disks. (Summary; " These operations include storage management services that allow **configuration changes** to be made dynamically to storage resources") [Examiner reminds Applicant that resources include virtual memory, stack calls, system resources, and data written on disk.]
> and in that it includes on the flow updating of the replicated resources by a dynamic introspection mechanism supplying the structure of the application to be replicated, as well as a dynamic graph of the resources and dependencies implemented.(Column 3; lines 33-36) [real time is on the flow]

Final Office Action dated March 19, 2008, pp. 2-3.

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983). In this case, each and every feature of the

presently claimed invention is not identically shown in the cited reference, arranged as they are in the claims.

Claim 40, as amended is as follows:

> 40. A computer implemented method for replicating a software application, the computer implemented method comprising:
> executing the software application on a primary node to form a master application;
> identifying resources and dependencies required by the master application to form required resources;
> updating the required resources dynamically on the primary node;
> generating a structure of the master application and a dynamic graph of the required resources from the required resources;
> replicating the resources by transferring the structure to a set of secondary nodes via a network to form a replica;
> restoring the replica on the set of secondary nodes to form a set of clone software applications;
> executing the set of clone software applications on the set of secondary nodes, without loss of context; and
> updating the set of clone software applications with incremental updates, of the required resources of the master application to create a hot standby application.

Kelkar does not anticipate claim 1 because *Kelkar* fails to teach the features as claimed. In particular *Kelkar* fails to teach the features of, "executing the software application on a primary node to form a master application," "identifying resources and dependencies required by the master application to form required resources," "updating the required resources dynamically on the primary node," "generating a structure of the master application and a dynamic graph of the required resources from the required resources," "replicating the resources by transferring the structure to a set of secondary nodes via a network to form a replica," "restoring the replica on the set of secondary nodes to form a set of clone software applications," "executing the set of clone software applications on the set of secondary nodes, without loss of context," and "updating the set of clone software applications with incremental updates, of the required resources of the master application to create a hot standby application."

With regard to the feature of, "executing the software application on a primary node to form a master application," applicants address portions of *Kelkar* cited by the examiner. The cited portions include Figure 2 and the following:

> A method, system, and computer program product to enable other nodes in a cluster to resume operations of a failed node. These operations include storage management services that allow configuration changes to be made dynamically to storage resources. Resource configuration data are synchronized on a set of nodes in a cluster immediately when a resource configuration change is made. If a node that has made a resource configuration change fails, the resource configuration change is available for use by other nodes in the set, each of which can resume operations of the failed node.

Kelkar abstract.

Kelkar apparently teaches managing storage resources for an application and not the application itself as stated above, "These operations include storage management services that allow configuration changes to be made dynamically to storage resources...the resource configuration change is available for use by other nodes in the set." *Kelkar* further appears to teach close to real time updating of resources as in:

> The present invention provides a method, system, and computer program product to make resource configuration information available to nodes in a cluster in as close to real-time as possible with minimal overhead.

Kelkar, col. 3 lines 33-36.

However, *Kelkar* in col. 5 lines 8-9 states, "nodes 110A and 110B are configured as servers for the same application program." *Kelkar* teaches management of configured resources for a single application and does not deal with configuring of the application itself. Further *Kelkar* does not teach cloning the application.

Kelkar teaches sharing resource configuration data and the configuration change is shared by setting attributes as in:

> Nodes 110A and 110B share resource configuration data, and each node has a respective copy, respectively labeled resource configuration data 370A and resource configuration data 370B.

Kelkar col. 5 lines 52-55.

> In the embodiment shown in FIG. 3, the configuration change is shared by setting attributes.

Kelkar col. 6 lines 28-29.

In contrast, the claimed features of the amended claim 40 are neither taught nor suggested by the disclosure of *Kelkar*. In addition, *Kelkar* fails to teach the features of "replicating the resources by transferring the structure to a set of secondary nodes via a network to form a replica," "restoring the replica on the set of secondary nodes to form a set of clone software applications," and "executing the set of clone software applications on the set of secondary nodes, without loss of context," because *Kelkar* manages configuration data change by sharing attribute settings as previously stated. *Kelkar* further teaches at col. 8 lines 15-17, "send the value of the attribute as the resource configuration data to the set of nodes."

Kelkar therefore does not teach "transferring the structure to a set of secondary nodes" because *Kelkar* does not create such a structure of the software application. Further *Kelkar* fails to teach forming a replica and restoring the replica because *Kelkar* teaches use of attribute values and sending attribute

values to another node. *Kelkar* teaches sending attribute values from one node to another node. In contrast the claimed feature creates a structure of a software application that is sent to another node where it is restored and executed. *Kelkar* fails to teach creation, transfer and restoration of such a structure. Further, *Kelkar* fails to execute the restored structure as is claimed because *Kelkar* deals only with attribute values of a resource.

In addition, *Kelkar* fails to teach the feature of, "updating the set of clone software applications with incremental updates, of the required resources of the master application to create a hot standby application." *Kelkar* teaches,

> In response to the failover, in action 6.3, cluster manager 330B activates resources. In one embodiment, cluster manager 330B calls an "online" entry point, or set of instructions, for each agent. Also in action 6.3, storage resource agent 416A, log agent 418B, and recovery agent 419B begin preparation to actively manage resources, such as storage resource 140. In action 6.4, each of storage resource agent 416A, log agent 418B, and recovery agent 419B requests resource configuration manager 360B for respective attributes. Resource configuration manager 360B obtains the attributes from storage resource attributes 476B, log attributes 478B, and recovery attributes 479B of resource configuration data 370B in action 6.5. Because resource configuration data 370A and 370B are synchronized, cluster manager 330B has access to current values for the attributes for storage resource 140.
>
> In action 6.6, resource configuration manager 360B provides storage resource attributes 476B, log attributes 478B, and recovery attributes 479B to the respective agents storage resource agent 416A, log agent 418B, and recovery agent 419B. In action 6.7, storage resource agent 416B uses storage resource attributes 476B to create storage group definition 140D on node 110B.

Kelkar, col. 8 lines 45-67.

Kelkar therefore teaches, responsive to a failover "storage resource agent 416B uses storage resource attributes 476B to create storage group definition 140D on node 110B." *Kelkar* therefore does not have a usable resource until the resource definition has been created and made ready. In contrast the claimed feature "updating the set of clone software applications with incremental updates, of the required resources of the master application to create a hot standby application." Therefore *Kelkar* does not teach the feature as currently claimed.

Kelkar fails to teach every element of a claimed invention as identically shown in that single reference, arranged as they are in the claims in accordance with the standard of *In re Bond*. Accordingly, *Kelkar* does not anticipate claim 40.

Because claims 41-57 depend from claim 40, the same distinctions between *Kelkar* and claim 40 apply equally well for claims 41-57. Additionally, claim 58 claims similar subject matter as claim 40, and is therefore likewise distinguished from the teaching of *Kelkar*. Accordingly, the rejection of claims 40-58 under 35 U.S.C. § 102 has been overcome.

II. Conclusion

The subject application is patentable over the cited references. Therefore, the subject application should now be in condition for allowance. Applicants invite the examiner to call the undersigned at the below-listed telephone number if, in the opinion of the examiner, a telephone conference would expedite or aid the prosecution of this application.

DATE: July 29, 2008

Respectfully submitted,

/Theodore D. Fay, III/

Theodore D. Fay, III

Reg. No. 48,504
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicants

TDF/wr